**API Access Practice Through WOKWI Simulation**

*Agatha Herma Putra*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: agathaputra24@student.ub.ac.id*

**Abstract**

This report presents the implementation of API access practice through WOKWI simulation. The study aims to explore the integration of APIs with WOKWI, a virtual microcontroller simulator, using PlatformIO for development, a database for data management, and Ngrok for external API access. The API was developed to facilitate communication between simulated hardware components and an external database. Ngrok was used to expose the locally hosted API, allowing real-time testing from external networks. The results indicate that WOKWI provides a reliable simulation environment for API testing, enabling seamless interaction between virtual sensors, actuators, and cloud-based data storage.

*Keywords : API, WOKWI, PlatformIO, Database, IoT Simulation, Web Development, Ngrok.*

**1 Introduction**

**1.1 Background**

API integration in IoT development is essential for enabling communication between hardware and software components. WOKWI, a web-based microcontroller simulator, provides an efficient platform for testing IoT applications without physical hardware. This study explores how APIs can be integrated into WOKWI simulations using PlatformIO, allowing data exchange between a simulated microcontroller and an external database. Additionally, Ngrok is utilized to create a secure tunnel, enabling API access from external sources for real-time testing and debugging. By implementing and testing API requests, this experiment aims to demonstrate the feasibility of virtual API interactions in an IoT environment.

**1.2 Purpose of the experiment**

The main objective of this experiment is to develop an API that facilitates communication between a simulated microcontroller in WOKWI and a cloud-based database. Ngrok is used to expose the local API to the internet, allowing external testing and integration with third-party services. This approach enables real-time data exchange, remote monitoring, and debugging capabilities for IoT applications. Additionally, the study aims to evaluate the effectiveness of WOKWI as a simulation tool for API-based interactions in an IoT environment.

**2.1 Methodology**

The API was developed using a backend framework to manage HTTP requests, store sensor data, and send control commands. The development environment was set up with PlatformIO, which was used to program a virtual ESP32 microcontroller in WOKWI. API endpoints were created to handle data transmission between the simulated device and the database. Ngrok was configured to expose the locally hosted API, allowing external devices and services to access it in real-time. Testing was conducted using API requests from WOKWI, Postman, and external applications to verify data accuracy. Finally, the performance of the system was evaluated based on response time, data integrity, and accessibility.

**2.2 Tools & Materials**

To successfully complete the API Access Practice Through WOKWI Simulation Using Laravel 11 and Ngrok, several tools and materials are required. These include:

1. Hardware Requirements
   * A computer or laptop with an internet connection
   * A web browser (Google Chrome, Mozilla Firefox, Microsoft Edge, or any other compatible browser)
2. Software & Online Platforms
   * Microsoft Visual Studio Code
   * Xampp
   * Postman
   * Ngrok
   * Laravel 11
   * Database phpMyAdmin
   * Wokwi
   * Platform.io
3. Additional Tools (Optional)
   * A text editor (such as Visual Studio Code or Notepad++) for working with GitHub repositories
   * A Git client (such as Git Bash or GitHub Desktop) for version control testing

These tools and materials ensure a smooth and efficient process for creating accounts and exploring the basic functionalities of both platforms.

**2.3 Implemention Steps**

Implementation for experiment making API Creation Practice Using Laravel 11 and Ngrok:

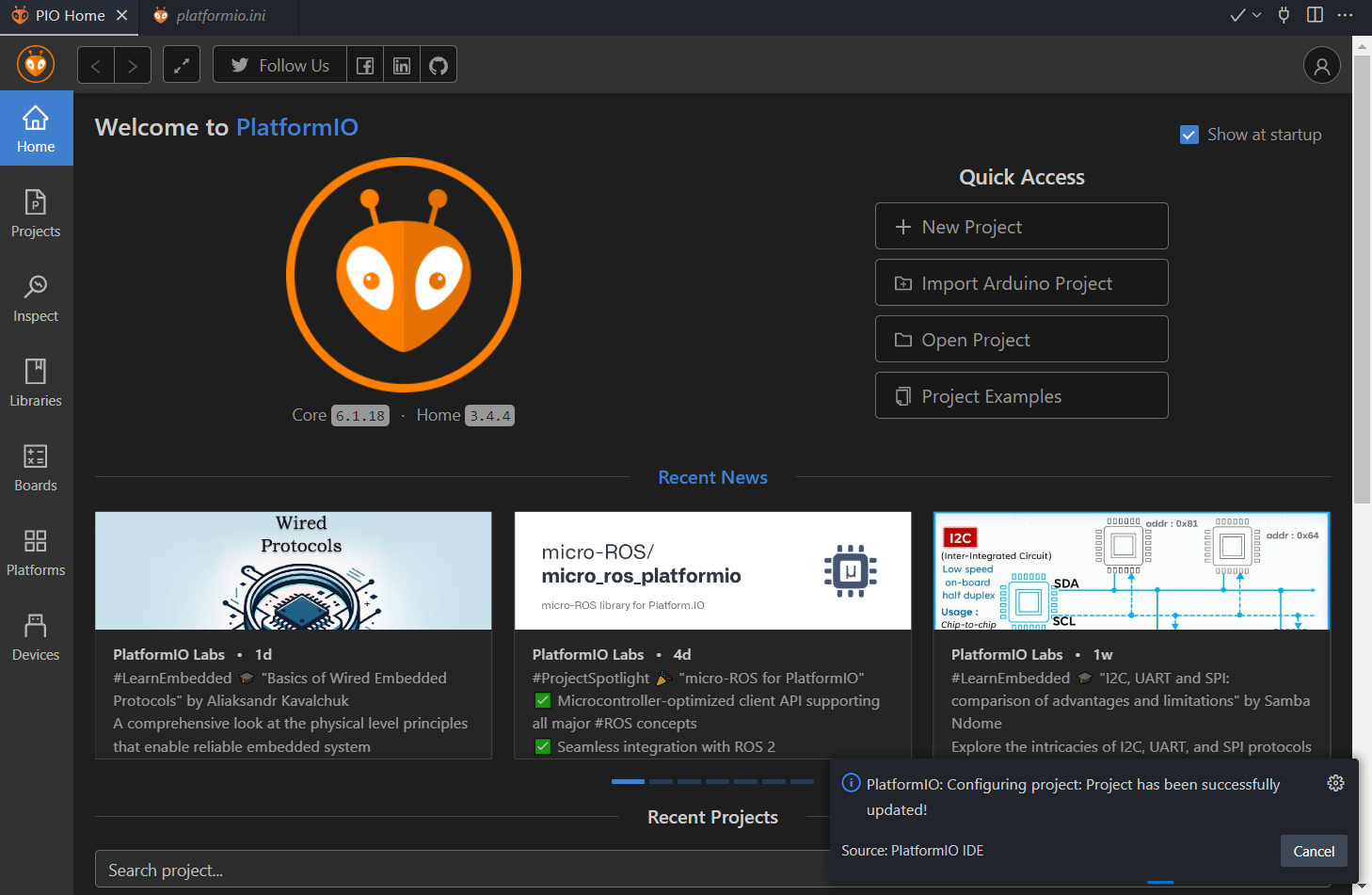
**1. Accessing the Platforms**

* Open a terminal on a Microsoft Visual Studio Code.
* Run the laravel API with the command

php artisan serve --host=0.0.0.0 --port=8080

**2. Create Project**

* Create a new Project wokwi simulator on platform.io



Make new Project

1. **Edit Project**

* Change main.cpp script

#include <WiFi.h>

#include <HTTPClient.h>

// Ganti dengan kredensial jaringan Wi-Fi Anda

// const char\* ssid = "Lab IT";

// const char\* password = "labit2024";

 const char\* ssid = "Wokwi-GUEST";

 const char\* password = "";

// URL lengkap server yang akan diakses

const char\* serverUrl = "http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts";

// Interval waktu antara setiap permintaan (dalam milidetik)

const unsigned long interval = 5000;

unsigned long previousMillis = 0;

void setup() {

  Serial.begin(115200);

  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan ke WiFi...");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println(" Terhubung!");

}

void loop() {

  unsigned long currentMillis = millis();

  // Periksa apakah interval waktu telah berlalu

  if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    if (WiFi.status() == WL\_CONNECTED) {

      HTTPClient http;

      // Inisialisasi HTTPClient dengan URL server

      http.begin(serverUrl);

      // Mengirim permintaan HTTP GET

      int httpResponseCode = http.GET();

      // Menampilkan kode status HTTP

      Serial.print("Kode status HTTP: ");

      Serial.println(httpResponseCode);

      // Menutup koneksi

      http.end();

    } else {

      Serial.println("WiFi tidak terhubung.");

    }

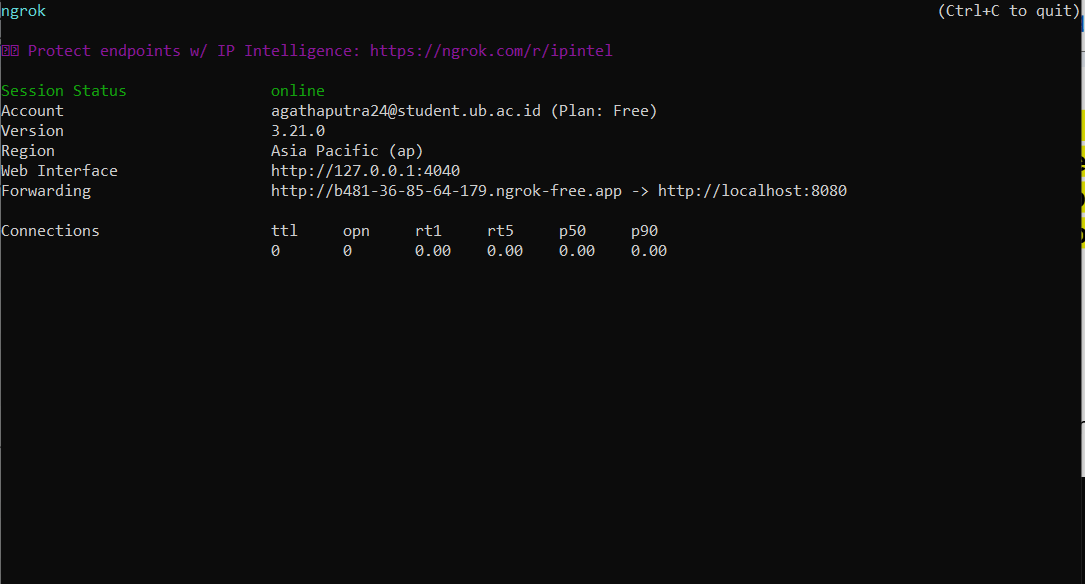
  }

}

* Run the ngrok server on the ngrok application with the following command code

**ngrok http --scheme=http 8080**

* Adjust the port address 8080 according to the port your Laravel application is running on.



Adjust the port address

* Add the wokwi.toml file and change its contents to be as follows

[wokwi]

version = 1

firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'

elf = 'C:\Users\mokor\Documents\PlatformIO\Projects\wokwi\_internet\.pio\build\esp32doit-devkit-v1\firmware.elf'

* Add the diagram.json file and change its contents to be as follows

{

"version": 1,

"author": "Uri Shaked",

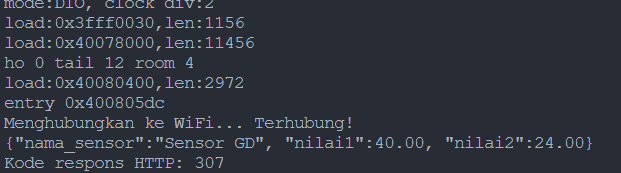
"editor": "wokwi",

"parts": [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} } ],

"connections": [ [ "esp:TX", "$serialMonitor:RX", "", [] ], [ "esp:RX", "$serialMonitor:TX", "", [] ] ]

}

* Start the simulator



Simulator Progress

* The next step is to modify the simulation by adding temperature and humidity sensors. The scenario is, the wokwi simulator will send temperature and humidity data to the API and save it to the mysql database as created in the previous chapter and Assemble the DHT22 sensor with ESP32 as in the example above. Then copy the diagram.json code to the diagram.json file in vscode.

{

    "version": 1,

    "author": "KAVITH BUDVIN",

    "editor": "wokwi",

    "parts": [

      { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 86.4, "left": 24.04, "attrs": {} },

      { "type": "wokwi-dht22", "id": "dht1", "top": 19.5, "left": -91.8, "attrs": {} }

    ],

    "connections": [

      [ "esp:TX", "$serialMonitor:RX", "", [] ],

      [ "esp:RX", "$serialMonitor:TX", "", [] ],

      [ "dht1:VCC", "esp:3V3", "red", [ "v19.2", "h67.2", "v-67.2" ] ],

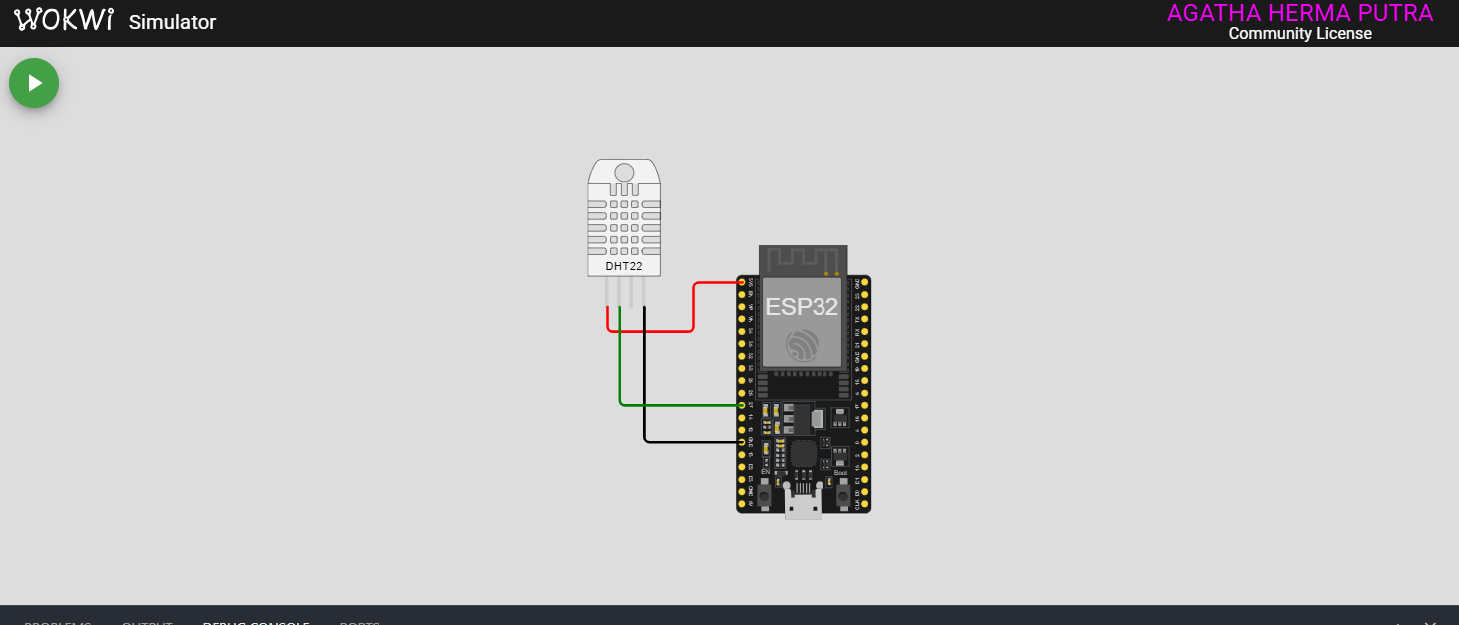
      [ "esp:GND.1", "dht1:GND", "black", [ "h0" ] ],

      [ "dht1:SDA", "esp:27", "green", [ "v0" ] ]

    ],

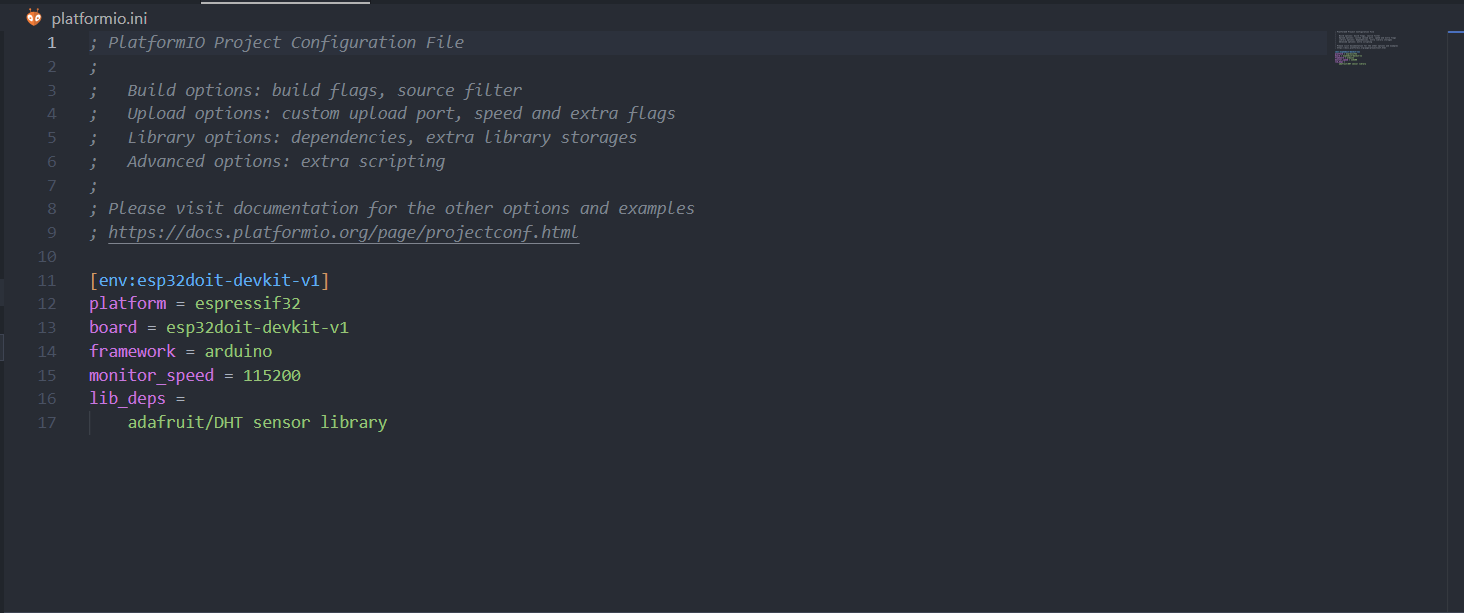
    "dependencies": {}

  }



Result

* Then change the platformio.ini file settings as follows:



Platform.ini

* Modify the main.cpp file

#include <Arduino.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include "DHT.h"

#define DHTPIN 27

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

// Ganti dengan kredensial WiFi Anda

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

unsigned long previousMillis = 0;

const long interval = 5000;  // Interval 5 detik (5000 ms)

void setup() {

  Serial.begin(115200);

  // Hubungkan ke WiFi

  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan ke WiFi");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println(" Terhubung!");

  dht.begin();

  // Tunggu sebentar agar koneksi stabil

  delay(1000);

}

void loop() {

  unsigned long currentMillis = millis();

  // Lakukan POST setiap interval yang telah ditentukan

  if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    float h = round(dht.readHumidity());

    // Read temperature as Celsius (the default)

    float t = round(dht.readTemperature());

    // Check if any reads failed and exit early (to try again).

    if (isnan(h) || isnan(t)) {

      Serial.println(F("Failed to read from DHT sensor!"));

      return;

    }

    // Compute heat index in Celsius (isFahreheit = false)

    float hic = dht.computeHeatIndex(t, h, false);

    // Inisialisasi HTTPClient

    HTTPClient http;

    String url = "http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts"; // Ganti dengan URL ngrok yang benar

    http.begin(url);  // Menggunakan HTTP, bukan HTTPS

    http.addHeader("Content-Type", "application/json");

String payload = "{\"nama\_sensor\":\"Sensor GD\", \"nilai1\":" + String(h) + ", \"nilai2\":" + String(t) + "}";

Serial.println(payload);  // Untuk melihat apakah payload sudah terbentuk dengan benar

    // Kirim POST request

    int httpResponseCode = http.POST(payload);

    // Tampilkan kode respons HTTP

    Serial.print("Kode respons HTTP: ");

    Serial.println(httpResponseCode);

    // Tampilkan respons dari server jika request berhasil

    if (httpResponseCode == 200 || httpResponseCode == 201) {

      String response = http.getString();

      Serial.println("Respons dari server:");

      Serial.println(response);

    } else {

      Serial.println("Gagal mengirim data");

    }

    // Tutup koneksi HTTP

    http.end();

  }

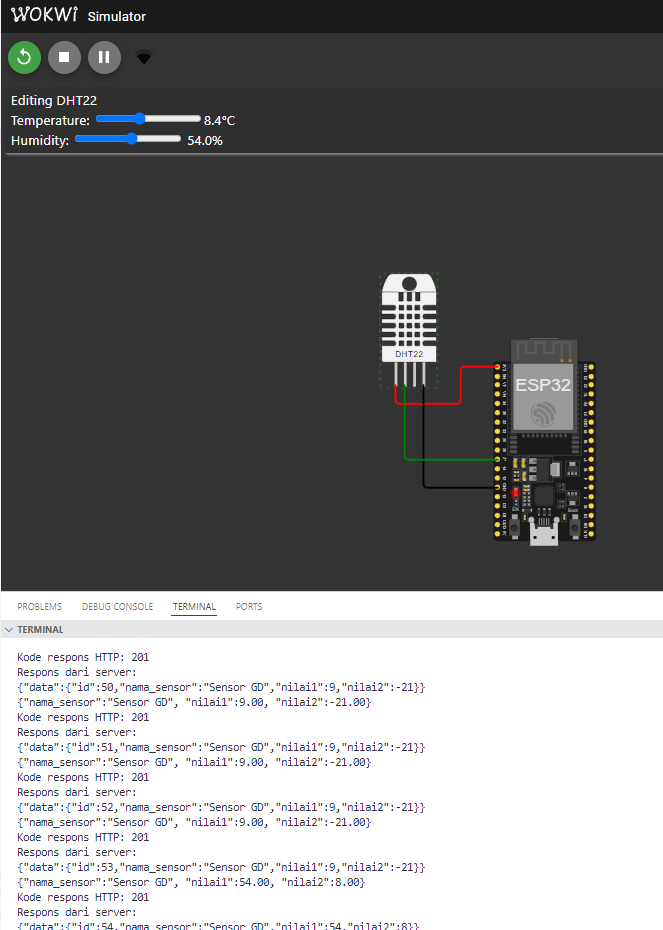
}

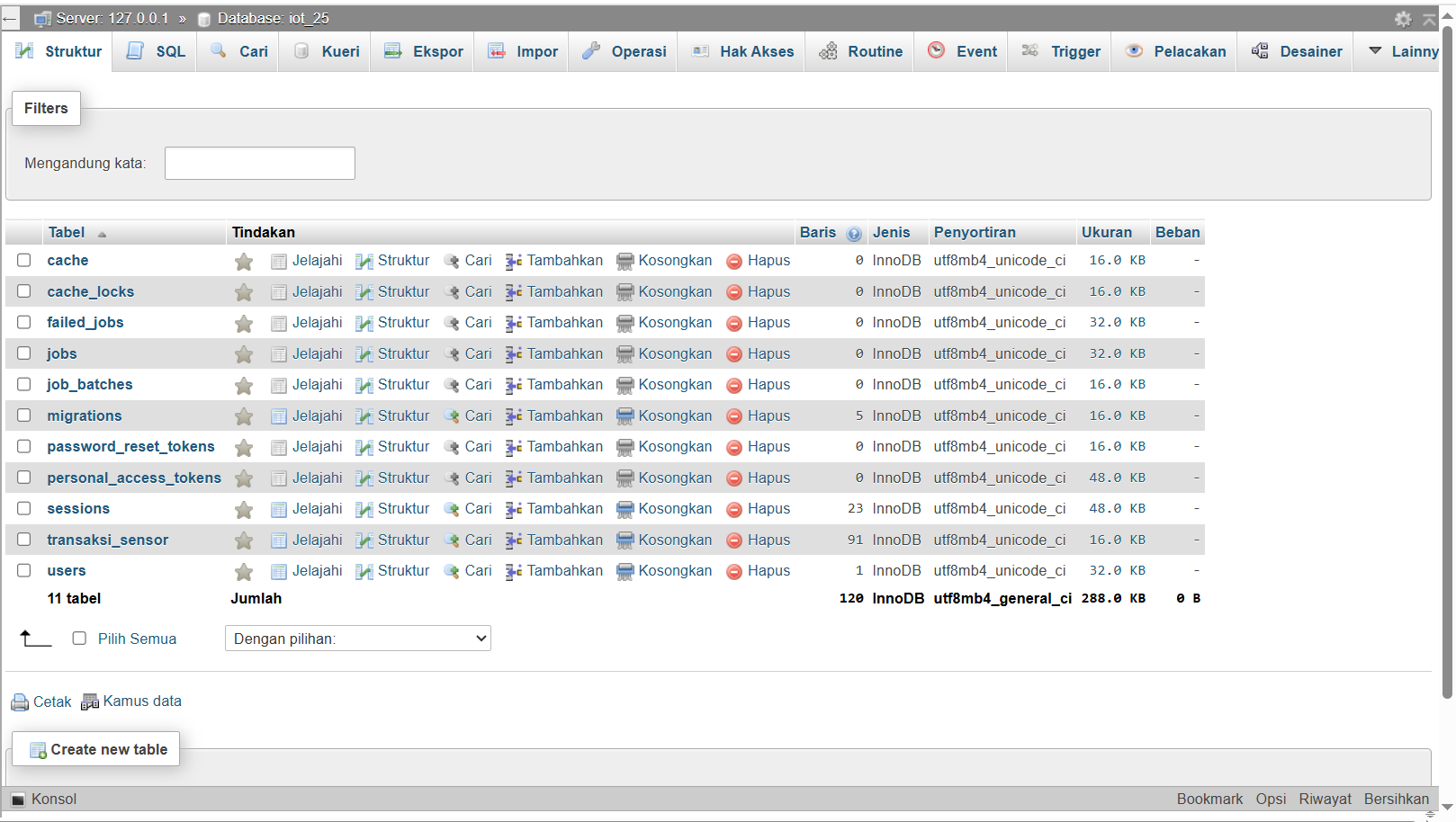
* Start The Simulator

**3. Results and Discussion**

**3.1 Experimental Results and Documentation**

* **Schema Development** The API was successfully designed and integrated with WOKWI, a cloud database, and Ngrok for external access.
* **Functionality Testing** API endpoints were tested using Postman and WOKWI, verifying correct data handling and external access.
* **Code Implementation** The WOKWI simulation successfully communicated with the API via Ngrok, sending and receiving data as expected.
* **Observation Results** The system functioned correctly, demonstrating real-time data exchange between the simulated ESP32, the API, and the database.
* **Evaluation** No major issues were found, but further optimization is required to enhance performance, improve security, and reduce latency in data transmission.





Database

1. **Appendix**

**A. Links to Official Websites**

The following links were used during the experiment for account registration and platform access:

* **Wokwi**: <https://wokwi.com>
* **GitHub**: <https://github.com>
* **Platform.**io
* **Ngrok**
* **Laravel**
* **Xampp**
* **mysql**

**B. Required System Specifications**

To ensure smooth execution of the experiment, the following system requirements were met:

* **Device**: Laptop with at least 4GB RAM
* **Operating System**: Windows 10 / macOS / Linux
* **Browser**: Google Chrome (Version 100+)
* **Internet Connection**: Stable with a minimum speed of 5 Mbps